Parallel identification of the spelling variants in corpora

Reynaert, M.W.C.

Published in:
Proceedings of the Third workshop on Analytics for Noisy Unstructured Text Data 2009

DOI:
10.1145/1568296.1568310

Publication date:
2009

Parallel identification of the spelling variants in corpora

Martin Reynaert
Induction of Linguistic Knowledge
Tilburg centre for Creative Computing
Tilburg University, The Netherlands
reynaert@uvt.nl

ABSTRACT

We present a new approach based on anagram hashing to globally handle the typographical variation in large and possibly noisy text collections. Typographical variation is typically handled in a local fashion: given one particular text string some system of retrieving near-neighbours is applied, where near-neighbours are other text strings that differ from the particular string by a given number of characters. The difference in characters between the original string and one of its retrieved near-neighbours we call a particular character confusion. We present a global way of performing this action: given a possible particular character confusion, we identify - in parallel, i.e. in one single operation on anagram-hash derived bit vectors - all the pairs of text strings in the text collection to which the particular confusion applies. The algorithm proposed here is evaluated on about 23,000 English attested typos from the Reuters RCV1 text collection. We further explore its usefulness for unsupervised linking of a historical Dutch word list to its contemporary counterpart.

Categories and Subject Descriptors

H. Information Systems [**H.3 INFORMATION STORAGE AND RETRIEVAL**]: H.3.1 Content Analysis and Indexing

General Terms

spelling variation, typographical, historical, OCR

1. INTRODUCTION

We present an approach to spelling variation detection and retrieval on the scale of large corpora. The final aim of this work is to able to take the word frequency list of a large corpus and to efficiently rid it of unwanted spelling variation up to a particular Levenshtein distance (LD) [5] limit.

Approximate matches are strings that are similar but not identical to the string one looks for. An in-depth overview of the state of the art in approximate string matching can be found in [7]. While many algorithms for finding approximate matches between word strings have been developed, so far no algorithm seems to have been put forward that, in a single parallel operation per character confusion, identifies the full set of approximate match candidates, i.e. all those word pairs that differ in exactly the same particular subset of characters, regardless of the actual character sequences. This is what we do in this paper. The approach takes a radically different tack from the usual local focus on one particular text string. We present a technique that allows for global identification, in parallel, of all the pairs of strings that differ in the same particular subset of characters. Given that a particular LD limit represents so many characters that are missing, were added or were replaced, the approach taken here aims at systematically identifying all the word form pairs that exhibit a particular confusion between so many characters in one single operation. We call this a global parallel operation, in contrast to the local sequential looking up procedures employed by other spelling correction systems.

In Section 2 we outline the sequential approach to spelling checking that underlies the subsequently proposed parallel version. Section 3 deals with a contemporary English corpus and the large collection of typos which were culled from it. We use this 23,000 attested typos gold standard primarily to test whether our parallel approach equals our sequential one in identifying the typos present. In Section 4 we perform a controlled experiment on historical spelling. We work towards our conclusions in Section 6 by discussing our findings and by taking a glance at related work in Section 5.

2. ANAGRAM KEY-BASED SPELLING CORRECTION

2.1 Sequential anagram key search

We adopt the core correction algorithm we described in depth in [9]. Anagram hashing first uses a bad hashing function to identify all word strings in the corpus at hand that consist of the same subset of characters and assigns a large natural number to them, to be used as an index. Informally, the numerical value for a word string is obtained by summing the code value, e.g. ISO Latin-1, of each character in the string raised to a power n, where n was empirically set at: 5. In effect, all anagrams, loosely defined as words consisting of a particular set of characters and present in the list, will be identified through their common numerical value. In the limit, associated to a particular key would

AND '09, July 23-24, 2009, Barcelona, Spain

be the $n!$ permutations given n distinct characters, if these were realized within the corpus. Given the set of characters **a, b, c**, there could be 3x2x1 = 6 permutations, but only two. i.e. 'abc' and 'cab' are likely to be encountered in an English dictionary. As the collisions produced by this function identify anagrams, we refer to this as an **anagram hash** and to the numerical values obtained as the **anagram values** (AVs) and **anagram keys**, when we discuss these in relation to the hash. Based on a word form's anagram key it thus becomes possible to systematically and sequentially query the list for any variants present, be they morphological, historical, typographical or orthographical.

The 'alphabet' used when the system is allowed to search up to three (and more) character edits, contains the AVs for single characters and all possible two-character and three-character combinations. This is called the AV-alphabet. Note that a single value in this alphabet represents a single character or any combination of two (2 combinations) or three (6 combinations) particular characters, allowing for efficient look-up. The focus word (FW) is not likely to contain all the characters in the AV-alphabet. The subset of values from the AV-alphabet derivable from the characters actually present in the focus word forms the FW-alphabet.

The lexicon contains the vocabulary of the validated dictionary (if any) as well as the vocabulary from the corpus to be cleaned. The lexicon is a regular hash built up at run-time having the AVs as keys and chained anagrams as values. The AV for the focus word and the FW-alphabet and AV-alphabet are used to query the lexicon for variants of the focus word. These variants can all be seen as variations and combinations of the usual error type taxonomy due to [2]. In the implementation, all four edit operations are handled as substitutions. For **substitutions**, a value from the FW-alphabet is subtracted and a value from the AV-alphabet added. A single query on the AV for 'yesterday' minus the AV for an 's', plus the AV for an 'a' may thus retrieve the typo: *yeaterday. **Insertions** are substitutions where a value from the FW-alphabet is subtracted and zero added. **Deletions** are substitutions where zero is subtracted and a value from the AV-alphabet added. To find **transposition** errors nothing needs to be added or subtracted, but the chained anagrams for the particular focus word AV need to be examined.

By systematically querying the lexicon hash all possible variants that fall within reach are retrieved. The actual reach is defined by the alphabet used, i.e. depends on whether the alphabet contains the AVs for single characters only, or also for the full set of character bigrams, or even trigrams. The actual number of hash look-ups required is defined by the number of unique values in the AV-alphabet and by the number of unique values for all the character combinations in the focus word.

In [10] we have presented pseudo-code for our implementation of this in TICCL, which stands for Text-Induced Corpus Clean-up.

2.2 From sequential search to parallel search

We call a specific difference between n specific characters between two strings a particular confusion. In electronic text produced by Optical Character Recognition (OCR), e.g. the character sequence 'in' is often misrecognized by the single character 'm'. This then constitutes an 'in-m' confusion. Sequential search through an anagram hash for this particular confusion involves checking for each key of the hash whether its counterpart, which has to fulfill the condition: focus word AV plus AV for 'in' minus AV for 'm', exists. We next describe how this sequential look-up can be turned into a parallel look-up, effectively reducing the n operations of the sequential look-out, where n is the number of keys in the hash index, into a single operation which identifies all the pairs: focus word/confusion counterpart.

How to turn TICCL into PARTICCL (Parallel Text-Induced Corpus Clean-up)? Imagine an old-fashioned wooden ruler. Imagine it is worn to the extent that the millimetre marks are no longer visible, only the larger 5, 10, 15, etc. millimetre marks are still discernible. Now imagine a second, even more worn ruler where through circumstances unknown only the 3, 13 and 18 millimetre marks are still there. If we cut off the front end of this ruler at the 3 mm. mark and line up both rulers, we see that the marks 10 and 13 and the marks 15 and 18 are perfectly lined up, i.e. they intersect.

In analogy to what we saw happening to the rulers, to make our sequential search in the anagram hash parallel, we map the hash values into a bit vector with the bits for each value set 'on', the intervening bits remaining 'off'. The second ruler's 3 mm cut-off is here the number of bits indicated by the anagram value which represents a particular confusion. In order to line up those bits set 'on' in both vectors for a particular confusion, we just truncate the second vector by the number of bits for that particular confusion and take the intersection. This simple process identifies all the members of the confusion set in one parallel move. If the Dutch definite article 'de' and its historical genitival form 'des' are at marks 10 and 15 of the first ruler and if the strings 'do' and 'dos' from the word list derived from a bad OCR-engine that is likely to recognize an 'e' as an 'o' are mapped on positions 13 and 18 on the second ruler, this process identifies the **pairs** 'de-do' and 'des-dos'. Both pairs are **members** of the **minimal character confusion** 'e-o', which in ruler terms would be expressed as '3 mm', but as n bits in the bit vector.

Parallel look-up is thus achieved by mapping the hash index into a sufficiently large **bit vector**, where sufficiently large corresponds to greater than the largest AV index key. For each key in the hash vector the corresponding bit in the vector is then set 'on'. In practice, after building the anagram key hash for the full list of word types encountered in the corpus, we set all the bits for the anagram keys present in the hash in a 32-bit unsigned integer bit vector. We use the Perl module Bit::Vector, thanks to Steffen Beyer,[1] for this purpose as it has all the necessary provisions for the bit vector manipulations we require.

We next **clone** this vector, i.e. make an exact copy of it. The copy will then be **shortened**, i.e. its n left-most bits are removed, where n is the number of bits corresponding to the anagram hash numerical difference between the characters in the particular confusion under consideration. In [9] the

[1] Available from: http://search.cpan.org/dist/Bit-Vector/.

anagram key values were obtained by raising the code page value for every character in a string to the fifth power and by summing the results for the string. This creates rather large anagram values. In this work we raise the values to the fourth power, allowing us to work with smaller bit vectors. In practice, for the 'in-m' confusion, this means that (AV for 'in' minus the AV for 'm' equals) 126,802,464 bits are deleted from the front of the bit vector clone. Finally, we take the **intersection** of the two vectors: the original anagram key vector and its truncated clone. The elements of the intersection give us the index references to the 'in-m' confusion pairs present in the anagram hash built for the corpus under consideration. Each set bit in the intersection vector gives us the reference to the first element in the pair, its particular value plus the confusion AV then gives the exact reference to the second element of the pair. These can then be straightforwardly be retrieved from the anagram hash, after which a necessary final LD check needs to be performed. This check is required because anagram hashing retrieves all anagrams associated with a particular key. A character confusion translates into a particular LD, which in this paper should not be larger than LD 2. Confusion counterparts that are retrieved for a particular focus word cannot exceed this limit and are to be discarded if they do because they are then anagrams of the actual true confusion counterpart. The fact that the chained anagrams have to be dealt with is in fact less costly than it may seem: there are not so many of them in an actual corpus. In Section 5 we further elaborate on this.

The parallel approach should deliver the same results as the sequential approach used so far. It is far more efficient because in order to find all the cases for a particular confusion, it does not need to iterate over all the keys of the anagram hash, but instead identifies all in its single intersection operation, however long the list actually is. Where the sequential approach requires additions and subtractions for identifying deletion and insertion errors, both are in PARTICCL identified in one go. An obvious limitation of the parallel approach proposed here is that it cannot find variants due to transpositions of characters only. These word strings get the same AV, so require sequential look-up. The total number of look-ups required for full coverage of all the combinatorial possibilities given single character confusions, 1 to 1 character confusions, 2 to 1, 2 to 2, 3 to 2, etc., confusions is dependent on the number of characters one regards as constituting the alphabet. Normalization of the alphabet, e.g. by upper- or lowercasing all alphabetic characters, brings huge reductions in the total number of required look-ups. But so does the abstraction over the actual ordening of characters within words afforded by the use of anagram values as the key to the words themselves. We can exhaustively examine all possible confusions between 1 and 2 character combinations in 63,150 intersection operations. These break down in 528 operations to examine all confusions involving insertion and deletion of a single character and substitutions involving two distinct single characters, 9,471 operations to examine all confusions between two distinct characters and a single character and 53,151 for all possible confusions between two distinct characters and two different distinct characters. Note that this amount of operations is orders of magnitude lower than the actual combinatoric possibilities offered by e.g. a 32 character alphabet.

Corpus	Lang.	Mb	Tokens	Types
R-RCV1	IE	714	134,031,130	1,626,038

Table 1: Reuters RCV1 Corpus Statistics: Corpus, language (IE: International English), size in Megabytes, number of word tokens, number of word types.

2.3 Output Filtering

The word pairs identified by the intersection of the bit vectors and retrieved from the anagram hash we call the **members** of the confusion set. Be advised that a particular confusion in fact describes a minimal confusion. The actual surface forms of a member pair may be very divergent: all we know a priori about them is that they differ by the set of characters implied by the confusion's anagram value and that the right hand of the pair will show the extra (or in anagram value terms: numerically greater) character(s). The actual sequence of the characters in the pair may be very different. This is where the LD comes in: only for the pairs retrieved need we measure their LD. If this measured LD exceeds the LD implied by the confusion's anagram value, the pair has been spuriously linked and should be discarded.

In our experiments reported in the next section we further only perform validated lexicon-based filtering of the output produced by the parallel confusion pair retrieval procedure. We chose the publicly available standard ISPELL US and UK (expanded and concatenated) dictionaries as our validated lexicon. Any pair returned for which both elements are present in the validated lexicon are simply discarded. This filters out the bulk of regular morphological variants, but necessarily fails for valid words and their morphological variants that are not in the validated lexicon.

3. TEST SET AND EVALUATIONS

3.1 A contemporary corpus and its typos: Reuters RCV1

In [9] we extensively studied the prevalence of typos in contemporary English newswire text from the Reuters RCV1 corpus [6], formally known as RCV1-V1. This corpus contains about 810,000 Reuters, English Language News stories. This study was limited to the word type list of word forms beginning with a lowercase character. In all, 33,488 word types were marked as being typos: erroneous variants of other words. This error list constitutes just more than 21% of the 159,085 items long type list, an unexpectedly high proportion. In terms of tokens, 33,488 typos means that the Reuters RCV1 corpus contains 1 erroneous form per 400 running words with a lowercased first character. The top three most frequently misspelled words and their corpus frequencies were *goverment [482], *milion [372], *occured [331].[2] Statistics regarding RCV1 corpus size and numbers of words are presented in Table 1.

For our evaluation purposes here we have further completed the list of typos culled from the Reuters RCV1 corpus by [9]. Of the 33K typos collected, 11K were there linked to their

[2] We present the corpus frequency of a word within square brackets.

Category	Levenshtein Distance LD						Total	%
	1	2	3	4	5	6-9		
deletion	10,138	333	19	31	2		10,523	36.25
insertion	8,255	245	26	9			8,535	29.40
substitution	3,748	117	6	1	1	1	3,874	13.35
transposition		3,528		5			3,533	12.17
multi-NC		590	141	26	6	2	765	2.64
multi-C		69	33	25	50		177	0.61
space deletion	1,209						1,209	4.17
space insertion	26						26	0.09
TOTAL	23,746	4,891	228	99	60	5	29,029	
%	81.80	16.85	0.79	0.34	0.21	0.02		100.0

Table 2: Statistics of the error categories in the 29,029 typo/correction list.

correct word form after visual inspection of their context. We have now likewise added the correct word forms for another 18K of the typos. In Table 2 we present the statistics of the RCV1 typo list in terms of the categories of errors encountered. First listed are the 4 simple categories of error, i.e. insertion, deletion, transposition or substitution [2]. Next we list multiple errors which cannot be described by reference to just one of the 4 categories of error alone. A multiple contiguous error (multi-C) would be the OCR-error *readmg for '*reading*, i.e. the multiple error consisting of deletion of an 'i' and substitution of the 'n' by 'm' is situated in one location within the word. A multiple non-contiguous error (multi-NC) would be the typo *momopology for *monopoly*. Note that the bulk of all naturally occurring, attested typos involve one or two edits, i.e. fall within LD 2: 98,5% in the case of the typos accounted for in the RCV1.

3.2 Preliminaries to the evaluation

We wish to measure how well the parallel implementation of our algorithm manages to resolve the typos to their correct counterpart as annotated in the RCV1 typo list. As we have stated, the parallel version cannot handle transposition errors, so these were left out of the accounting. Neither do we here address the 4% of errors concerning space deletions and insertions. This we chose to defer to later work and so these cases were also left out of the accounting. In all, we evaluated our system on 23,063 typos.

It is a moot point, for the purposes of the evaluations performed here, whether or not we have actually managed to assign the correct resolution to a particular typo in the RCV1 list. The typo *toaday was resolved as *today*, rather than as *toady*. If the word *toady* were to be present in the RCV1 (it is not), an exhaustive run of our system would link this particular typo to both possible resolutions, on the basis of the same confusion, i.e. one added 'a'. As we wish to perform a glass-box evaluation [8] of the capabilities of the proposed correction mechanism, we want to measure the resolution of a particular typo to one of its possible resolutions rather than to measure how many possible resolutions for a particular typo there are within the limits set and within the limits imposed by the particular corpus used. We presented our system with the full list of 159,085 RCV1 word types beginning in lower-cased characters as described in Section 3. On the basis of this list, containing the word types only, we also measure for how many of the correct word types the system reports having found 'corrections', i.e. False Positives. This, at least, gives us an idea of the amount of work still waiting us: finding ways and strategies for avoiding incurring too many False Positives.

We evaluate in terms of recall and precision, resulting in the combined F-score [12]. These metrics are derived from the numbers of True Positives (TPs), False Positives (FPs) and False Negatives (FNs) returned by the system. **True Positives** are defined by what constitutes the **target** of our exercise. The target is the non-word variants present in the RCV1 corpus-derived list to be processed. **False Positives** are non-word variants or real words, that are erroneously reported to be variants for a particular focus word. **False Negatives** are those items in the list of known, annotated variants for the particular focus word that are absent from the list of variants returned for this focus word, i.e. that the system was not able to retrieve or 'correct'. The formulae used are as follows:

$$\text{Recall} = \text{R} = \frac{TP}{TP+FN} \qquad \text{Precision} = \text{P} = \frac{TP}{TP+FP}$$

Since we deem recall and precision to be equally important, the harmonic mean of R and P, the simplified F measure, F, is given by:

$$\text{F-score} = \text{F} = \frac{2 \times R \times P}{R+P}$$

3.3 Evaluation results

We ran the system 5 times, taking into account progressively longer words only. So, at word length 5, we only take into account words of length 5 and higher. Results are presented in Table 3. The sum of True Positives TP and False Negatives FN gives the total amount of variants on which we evaluated per word length and higher. Also listed are the numbers of False Positives FP incurred. We list the Recall R, Precision P and F-scores F per word length and higher.

We see that our parallel look-up procedure retrieves nearly 100% of the real-world, attested typos in the gold standard. Logically, it should achieve 100%. It did not because of a small class of 'overlooked' confusions at LD 2: we did not look for confusions concerning one alphabetical character and one non-alphabetical character, e.g. *atr=tributed for

				Overall Score			Score at LD 1			Score at LD 2		
L	TP	FN	FP	R	P	F	R	P	F	R	P	F
2	22953	110	149468	0.995	0.133	0.235	0.996	0.647	0.785	0.979	0.007	0.015
5	22136	93	85538	0.996	0.206	0.341	0.997	0.712	0.831	0.979	0.013	0.025
10	9225	13	11038	0.999	0.455	0.625	0.999	0.814	0.897	0.988	0.060	0.114
15	493	2	414	0.996	0.544	0.703	0.998	0.773	0.871	0.979	0.142	0.249
20	40		11	1.000	0.784	0.879	1.000	0.857	0.923	1.000	0.444	0.615

Table 3: Overview of the performance scores by word length (L). For each length, the words of that and higher length were considered. Presented are True Positives (TP), False Negatives (FN) and False Positives (FP). We further list Recall (R), Precision (P) and the F-score (F), calculated for both LDs 1 and 2 (Overall Score), and at LD 1 and 2, respectively.

attributed. Further, it failed to propose corrections, due to the validated lexicon filtering, for some confusables such as the very infrequent real word *wold* for *would*.

We report both recall and precision figures, although - for the purposes of this paper - we deem the former far more relevant. We believe that precision should be stated to give at least an idea of the amount of false positives incurred by the approach. However, the aim here is to show that parallel confusion pair look-up over an entire corpus is feasible. We believe our recall scores clearly show that it is. The precision of the present algorithm can vastly be improved by incorporating some simple rules, either hand-crafted on the basis of available grammatical knowledge of the language, or text-induced on the basis of the actual observed numbers of confusion pairs returned by the system. We defer that to future work, but take a definite step in the direction of making use of the obtained global statistics in the next section.

For some words, there simply are no other words resembling them to the extent that these would fall within the LD of 1, 2 and even more edits. As such, words with a higher neighbourhood density [3], especially short words and words derived from a stem and highly common pre- and/or affixes, are far more likely to incur more False Positives. The precision scores on LD 2 clearly illustrate this and call for a stricter LD limit on the shorter words.

In the above, we have established that we have, provided one checks for all necessary confusions, a system that may achieve perfect recall. Remark that the scores imply that PARTICCL manages to resolve the Multi-C and Multi-NC errors at LD 2 for which we provided the statistics in Table 2. We have seen that this entails low precision if the necessary steps to filter out spuriously linked word pairs are not taken. We refer the reader to [9] and [10] for more detailed information on which filtering strategies can be applied in order to enhance precision.

In the next section we apply PARTICCL not to spelling correction, but to the related problem of identifying historical spelling variants for contemporary word forms. This section should be seen as an experiment under controlled circumstances. We intend to use PARTICCL on historical, OCR-ed corpora and hope to be able to make good use of the character confusion statistics it provides for more informed post-correction. We test it here on a corpus containing historical variation only and study what the statistics tell us.

4. HISTORICAL AND CONTEMPORARY SPELLING VARIANTS

One objective of our work is to find ways of automatically linking historical word forms to their contemporary canonical form, in the present case for Dutch. We here study whether the global statistics obtainable by PARTICCL can help this endeavour. To this end we feed the algorithm the contemporary Dutch word list, further referred to as GB05 as is available in the last versions of what is known in the Netherlands as 'the Green Booklet'[13][3]. A predecessor of this, further referred to as GB14, the 1914 seventh edition of the 1865 word list compiled by de Vries and te Winkel, we obtained from Project Gutenberg[4]. There are some limitations on this work imposed by the composition of the historical and contemporary word lists. Both lists have not been built according to the same specifications. At least the contemporary list is not a complete word list of Dutch today. It is geared, according to the preface, to 'those words that likely pose spelling problems'. The historical and contemporary word lists are not in any way aligned and show some considerable divergence in their vocabulary. GB05 has 207,738 word types, GB14 has 102,844. They share 58,879 types, allowing for 251,701 unique word types in all.

Though we have some knowledge of the spelling changes that were imposed by law in the Netherlands and Flanders in the late nineteen forties, we do not have a well-defined set of rules of the official spelling reform. We doubt if such a rule set exists and know there are large numbers of exceptions to such rules. The objective of this exercise is to build a historical-contemporary lexicon for Dutch. We would like to achieve this, as far as possible, in an unsupervised and automatic way, using no prior knowledge about either of the spelling systems and the differences between them. The approach taken here should to the largest possible extent be applicable to other languages or language varieties.

In effect we here build a full mapping of all the transitions between all the words in both the contemporary and historical word lists. I.e. given a contemporary word form, by changing at most two characters anywhere in the word, obtain all the word forms in the historical list that are in fact the result of these character changes. To cut down on processing we use a simplified alphabet. We assume here that no historical changes occurred in the use of diacritics other than use versus non-use. This allows us to trim down the

[3]Available from TST-Centrale: http://www.inl.nl/
[4]http://www.gutenberg.org/files/22722/22722-8.txt

alphabet we use for processing: we replace all occurrences of diacritics by an otherwise unused character, e.g. '2'.

As a first step, we have PARTICCL retrieve all the members of each possible confusion set given combinations of one and two characters. Retrieving the members for these 10,201 confusions in fact produces far more spurious links than links that are in fact the result of the spelling changes, in line with what we have seen for the RCV1. The task we face is to remove all these spurious links and retain only those that constitute in actual fact a historical-contemporary spelling word pair. We limit the present exercise to single and contiguous double character changes. We do not here examine the 2 to 2 character confusions, we know of only one historical spelling change to which this applies: 'qu' (as in: 'qualiteit', E: quality) versus the 'kw' (as in: 'kwaliteit'). This allows us to filter the retrieved pairs on the basis of the actual occurrence of a particular character sequence within the retrieved pairs, allowing us to discard the absolute bulk of spuriously retrieved pairs. This filtering complements the filtering on LD between the two elements of the pairs.

For the purposes of discovering the historical spelling changes, we fill the first bit vector with the hash key values representing the contemporary word list GB05. The second vector is not a clone of this, but is filled with the values from the historical word list GB14. In practice, we reuse the second vector, truncating it repeatedly by the number of bits indicated by the numerical difference between the hash key value of the current character confusion (the one we will presently focus on) and on the value of the previous one, some examples of which are given in Table 4. This presupposes that the confusions to be examined are presented to the system in ascending numerical order.

A first filtering after the word pairs for a particular confusion have been retrieved is based on the co-occurrence of the retrieved historical element of the pair in the contemporary list. A 'historical' word form that persists today, is not a historical form.

We next see if the minimal confusion can be expanded to perhaps more meaningful or distinctive patterns. This is done per confusion member set: we collect all the character bi- and trigrams from each member pair that match the characters of the minimal confusion. We keep count of all these matching bi- and trigrams over the full member set. The top n bi- and trigrams are subsequently used to construct the expanded pattern sets and for each set the matching member pairs are finally collected and counted. We present an overview of this for the minimal confusion 'A' on our data in Table 5.

In all, PARTICCL returned 9,173 expanded patterns for 2,760 minimal confusions. The bulk of the 10,201 confusions were thus found not to have any members. This process could therefore easily be made more efficient by taking into account corpus derived character ngram-statistics: it does not make sense to look for variants based on character ngrams that do not or very rarely occur. We summarize our findings concerning the top 14 confusions, i.e. those for which most members were observed, in Table 6. We see that PARTICCL identifies both recurrent morphological patterns as well as consistent historical spelling changes. For automatic identification of the former, we assume that it would be sufficient to incorporate e.g. the morphological information available for a great many languages in the affix files that come with the Ispell dictionaries. Conversely, the system might be used to identify morphological patterns not available in the affix file for a particular language. The fact that morphological patterns prop up in our results is both a consequence of the fact that in the preprocessing of the GB files we in fact discarded available information about the morphological relatedness of some word forms and of the fact that both lists contain different information, GB05 does not in fact list regular comparatives and superlatives.

The confusions classified as 'diffuse' exhibit, largely depending on the particular expanded pattern, a mix of spuriously linked word forms and bona fide contemporary/historical word pairs.

It can be seen that clear patterns emerge even from these top 14 confusions: historical Dutch exhibited a doubling of vowels, in fact long vowels were usually spelled with a double vowel. After the spelling reform in the late forties long vowels in open syllables were no longer written with a double vowel. There was some simplification, here exemplified by the (partial) disuse of 'sch' for the 's'-sound (E: French was 'Fransch' in GB14, is 'Frans' in GB05, but E: Russian remains: 'Russisch') and of 'ph' for the 'f'-sound (E: philosopher was 'philosoof', is now: 'filosoof'). It also emerges some consonants were no longer written doubly.

We finally had to decide not to attempt to measure the accuracy of this test. We started out by randomly sampling 10% from each set of confusion members. We were soon defeated by insufficient knowledge of the historical vocabulary, compounded certainly by the absence of clarifying contexts, leaving us in doubt too often about whether or not to assign a 'yes' or 'no' to a particular pair. We think this is as far as we can take this in an unsupervised manner. At this point, partly due to the current unavailability to PARTICCL of morphological information, it is up to lexicographers to finish the job, guided by the statistics gathered here. Their job should be greatly facilitated by the ordered lists per known character confusion that PARTICCL delivers.

5. DISCUSSION: USES, POSSIBILITIES, RELATED AND FUTURE WORK

We have in the previous section demonstrated that the minimal confusions represented by their anagram values can automatically be expanded into more meaningful patterns for which statistics can be gathered in the same run and that the amount of members for a particular confusion set gives an indication of the confusion's relative importance.

PARTICCL in fact performs an exhaustive search over the possible permutations given a particular subset of characters that happen to have been realized in the particular corpus it is set to work on. This is less expensive than it may seem in that in practice within a language only a limited number of valid word forms are realized. For the RCV1 list with the attested typos removed there are only on average 1.046 anagram collisions chained to the anagram keys.

GETDIFF: 12095 [O2-R or 2O-R] minus [] = 12095
GETDIFF: 21056 [G2-K or 2G-K] minus 12095 [O2-R or 2O-R] = 8961
GETDIFF: 103842 [QE-Z or EQ-Z] minus 21056 [G2-K or 2G-K] = 82786

Table 4: TICCL debugging output showing the calculation of the numerical differences between subsequent confusions to be examined. Alphabet used here: characters 'A-Z' and '2' for diacritics.

AV	minimal confusion	expanded pattern	exp. pattern members	confusion members	%
17850625	-A	E-AE	39	62	62.90
17850625	-A	A-AA	15	62	24.19
17850625	-A	R-RA	14	62	22.58
17850625	-A	P-PA	7	62	11.29
17850625	-A	T-AT	3	62	4.84

Table 5: Statistics gathered by PARTICCL for the minimal confusion 'A' (i.e. extra somewhere in words in GB14 in comparison to the corresponding words in GB05). Interestingly, the first two ranked expanded patterns both identify historical spelling changes, e.g. aesthetica vs. current 'esthetica' (E: aesthetics) and 'kongeraalen' vs. 'kongeralen' (E: conger eels).

We fully intend to employ the algorithm introduced here in work on very large historical, automatically digitized text collections such as are being produced around the world in numerous large libraries. Historical collections bring their challenges in historical spelling variation. OCR-ed collections bring their even greater challenges due to the vagaries of the process. We think that using our algorithm, studying first what variation is actually present in a given collection, will give a clear indication of which major confusions to tackle first and foremost. Tackling a known confusion on the basis of the acquired statistics about its prevalence should in terms of the actual tokens in a corpus allow for greater recall and far greater precision than has been possible so far.

Another possible use is query expansion in Text Retrieval, which might be helped by the typographical variants being linked to possible query terms. The algorithm may also prove valuable in cognate detection between two or more languages. Given e.g. the cognates 'facility' and the Spanish 'facilidad', the difference would reduce to the character bigram 'ty' and the character trigram 'dad', which may prove to be recurrent and therefore useful for extracting all analogical cognates from an English-Spanish corpus in a single operation.

Approaches to spelling correction have been proposed which work incrementally [1], i.e. based on a small LD difference between a particular confusion pair, search for more elaborate confusions between the correct form and another incorrect form by way of another small LD step restarting from the first identified incorrect form. This relies on the intermediate steps being present. Given that we now have the means to exhaustively gather all the pairs displaying a particular confusion, we are now able to evaluate whether this assumption holds and to what extent it does. If this holds, this may mean that an exhaustive run for all confusions on LD 3 is not warranted, in so far that the combined results of the far less expensive exhaustive LD 1 and 2 run may nevertheless deliver higher LD variants as well, incrementally. We hope to study this in the context of OCR-ed corpora, where far more higher LD errors are to be expected.

Our approach achieves far better recall than the systems developed in the Finite State paradigm by e.g. [11]: about 66%[5] and more recently based on an FSA with in-built LD capabilities: about 90% in [4] . We make no claims towards good precision in this paper, the focus being on exploring rather than exploiting a parallel approach to spelling variation.

The algorithm we have presented is not language-dependent in se. In [9] we worked on both English and Dutch and built a trilingual spelling correction system by further adding French to a mixed English-Dutch system. Our parallel system retains this feature.

Another attractive feature of our parallelization of spelling variant retrieval is the fact that the search for particular confusions can easily be distributed over as many processors or computers one has at hand. So the parallel look-up in actual fact enables easy parallelization of the full task. This should enable the method to scale to the largest corpus sizes. We have run PARTICCL on four processors for this work by simply dividing the list of the anagram keys for the 10,201 confusions to be examined in four equal parts. This ensures there is no overlap between the four systems running independently and that no double work is done. Doing all the work for the experiments on Dutch required about 8 seconds per confusion.

This work nevertheless raises a number of questions which we have not even tried to address. One is the complexity of the process. We have given a minimal indication of the cost in time. We have not compared this cost to what it would be if all the work were done sequentially. Even when everything is done sequentially, the locally obtained information could also be gathered to build the global picture. We think the parallel approach may outperform the sequential one if the scale of the task is far greater than comparing two wordlists, e.g. when working on a decade's worth of newly digitized newspapers. The gain would lie primarily in the fact that one can gather a global picture of the full variation present

[5]We sincerely apologize to the authors for having inadvertently misrepresented their scores in the printed version of [10].

patterns	members	min. confusion	top exp. pattern	classification	examples (left= contemp. right = histor.)
7	1486	-E	E-EE (94.15%)	Historical	REGERING-REGEERING (E: government)
5	1245	E-ST (etc.)	E-ST (89.48%)	Superlatives	VRIJE-VRIJST (E: free-most free)
5	1110	-O	O-OO (90.27%)	Historical	OGEN-OOGEN (E: eyes)
5	1051	-R	E-ER (72.12%)	Comparatives	HANIGE-HANIGER (E: cocky-cockier)
5	972	-CH or -HC	S-SCH (96.81%)	Historical	VIS-VISCH (E: fish)
2	238	E-CH (etc.)	SE-SCH (95.80%)	Cont. - Hist.	VERSE-VERSCH (E: fresh)
5	193	-D	E-DE (84.97%)	Historical	BLIJER-BLIJDER (E: happy-happier)
6	158	-ER or -RE	-ER (36.71%)	Diffuse	Analagous to E: work - worker
5	157	F-HP (etc.)	F-PH (100%)	Historical	MORFINE-MORPHINE
5	142	-OR or -RO	-RO (11.97%)	Diffuse	STOMEN≠STROOMEN (E: to steam - streams)
5	123	-N	N-NN (24.39%)	Diffuse	REVOLUTIONAIR-REVOLUTIONNAIR
5	111	ON- or -NO	-ON (33.33%)	Diffuse	KOPEN≠KNOOPEN (E: to buy vs. to knot) vs. GAZELLEOGEN-GAZELLENOOGEN (E: doe-eyes)
5	108	-NE or -EN	-EN (37.04%)	Diffuse	BOONSTAAK-BOONENSTAAK (E: bean stalk)
5	105	-S	L-SL (23.81%)	Diffuse	KABELLENGTEN-KABELSLENGTEN (E: cable lengths) vs. MARCHEN-MARSCHEN (E: marches)

Table 6: Overview of the top 14 confusions returned by PARTICCL in its comparison of GB14 and GB05. Spuriously linked pairs are marked by '≠'

prior to starting to do all the further processing on every single member of a particular character confusion. One could then decide to limit that work to just the largest member sets.

6. CONCLUSIONS

In this paper we have presented a global approach to tackling spelling variation in corpora. We have also presented some new data on typos in a large English corpus. We have proposed a parallel look-up algorithm for identifying all the pairs of words that happen to be confused in particular characters. We have demonstrated that we can exhaustively examine the word type list of a large English corpus for all the character confusions up to LD 2. We have illustrated that this works with near perfection as regards recall for over 23,000 attested typos. We have thereby shown that parallel spelling variant retrieval works as advertised and that scalability is ensured because of the inherent distributibility of the character confusion approach. Results on discovering variation patterns automatically and on the possible uses of global statistics on the corpus spelling variation gathered in the process are highly encouraging, but require more formal evaluation on historical OCR-ed corpora.

7. ACKNOWLEDGMENTS

TICCL and PARTICCL prototypes were developed within an NWO Exact Sciences Hefboom project. The production version of TICCL was commissioned by the Koninklijke Bibliotheek - Den Haag. We further thank our anonymous reviewers for their valuable comments.

8. REFERENCES

[1] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In D. Lin and D. Wu, editors, *Proceedings of EMNLP 2004*, pages 293–300, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[2] F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, Volume 7, Issue 3 (March 1964):171–176, 1964.

[3] U. Frauenfelder, R. Baayen, F. Hellwig, and R. Schreuder. Neighbourhood density and frequency across languages and modalities. *Journal of Memory and Language*, 32:781–804, 1993.

[4] A. Hassan, S. Noeman, and H. Hassan. Language independent text correction using finite state automata. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, 2008.

[5] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Cybernetics and Control Theory*, volume 10(8), pages 707–710, 1965. Original in: Doklady Nauk SSSR 163(4): 845–848 (1965).

[6] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[7] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[8] M. Palmer and T. Finin. Workshop on the evaluation of natural language processing systems. *Computational Linguistics*, 16(3):175–181, 1990.

[9] M. Reynaert. *Text-Induced Spelling Correction*. PhD thesis, Tilburg University, 2005.

[10] M. Reynaert. Non-interactive OCR post-correction for giga-scale digitization projects. In *Proceedings of CICLing 2008. Lecture Notes in Computer Science Vol. 4919/2008*, pages 617–630, Berlin / Heidelberg, 2008. Springer.

[11] C. Ringlstetter, K. U. Schulz, and S. Mihov. Orthographic errors in web pages: Toward cleaner web corpora. *Computational Linguistics*, 32(3):295–340, 2006.

[12] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1975.

[13] Woordenlijst Nederlandse Taal. *Samengesteld door het Instituut voor Nederlandse Lexicografie in opdracht van de Nederlandse Taalunie*. SDU Uitgevers, Den Haag, 2005.